SPECIAL SECTION

Computational Metrology

Theodore H. Hopp
Factory Automation Systems Division, National Institute of Standards and Technology, Room A127, Building 220, Gaithersburg, MD 20899-0001; hopp@cme.nist.gov

AS PART OF ITS GOAL TO DEVELOP THE BASIS FOR A U.S. NATIONAL STANDARD FOR CMS SOFTWARE PERFORMANCE EVALUATION, THE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST) IS IMPLEMENTING A SPECIAL TEST SERVICE TO BE OFFERED THROUGH THE NIST MEASUREMENT SERVICES PROGRAM. THIS SERVICE IS BASED ON A BLACK-BOX SOFTWARE MODEL, IN WHICH THE INTERNAL STRUCTURE OF THE SOFTWARE AND THE CHOICE OF SOLUTION METHODS ARE ASSUMED TO BE UNKNOWN. THE MODEL IDENTIFIES A NUMBER OF ERROR SOURCES FOR DATA ANALYSIS SOFTWARE. THE AUTHORS ARE DESIGNING TEST METHODS FOR IDENTIFYING THE VARIOUS COMPONENTS OF THE MODEL AND HOW THOSE COMPONENTS RELATE TO MEASUREMENT UNCERTAINTY IN INSPECTION APPLICATIONS.

Data analysis software has become increasingly important in modern dimensional measurement systems. This is particularly true of coordinate measurement systems (CMSs) such as vision systems, theodolites, photogrammetry, and coordinate measuring machines (CMMs). Despite the obvious benefits of using such software, computations to convert raw data to reported results can be a major source of error in measurement systems. Yet there are no standards or accepted methods for evaluating the effects of software on the overall uncertainty of measurements. The term *computational metrology* refers to the study of the effects of data analysis computations on the performance of measurement systems. This article identifies those aspects of computational metrology that are particular to CMSs. In particular, we will examine the objective of the computations, how these functions are carried out in software, and how software performance can be tested.

TOLERANCING & METROLOGY
PART I
THEORY

Not everyone shares the view that computational metrology is a significant area of study. For example, an ASME standard on measurement uncertainty states:

> *Computations on raw data are done to produce output (data) in engineering units. Typical errors in this process stem from curve fits and computational resolution. These errors are often negligible [1].*

The standard does deal with the propagation of errors through computations, but the above is the only mention of computations as a source of errors.

During the last few years, however, much evidence has been discovered that data analysis can be a significant source of errors. In the mid-1980s, Germany began a program of testing CMM software, with the express purpose of improving what was perceived as low quality of commercial fitting algorithms [2]. In 1988, Walker issued an advisory in which he reported the results of experiments with commercial inspection systems:

> *Certain algorithms . . . are capable of stating that the measurement is worse than the actual data gathered up to an error of 37 percent and that the measurement is better than the actual data gathered up to an error of 50 percent [3].*

And in 1989, Estler analyzed a measurement device for inspecting the solid rocket boosters casings for the NASA space shuttle [4]. He reported that the data analysis software was the single largest source of error in the entire system.

There is a great deal of ongoing research on CMS algorithms. (See [5] for a survey.) There is also growing interest by government standards laboratories in testing the performance of CMS software, particularly in Great Britain [6] and Germany [7]. Both countries offer services to test CMS software by comparing results for test data sets to results obtained from reference software. In the U.S., NIST is developing a similar service [8].

Of particular interest is the performance of software for calculating geometry that best fits a set of points measured on a part surface. Fitting is at the core of most measurements made by CMSs.[1] From a metrological point of view, two factors determine fitting software performance: 1) the choice of fitting objective and 2) the quality of the software implementing that objective.

1. Fitting is usually thought to involve optimizing the fit between a single geometric form—plane, cylinder, etc.—to a data set. However, so-called soft functional gaging can also be thought of as fitting a solid model of a functional gage to the data.

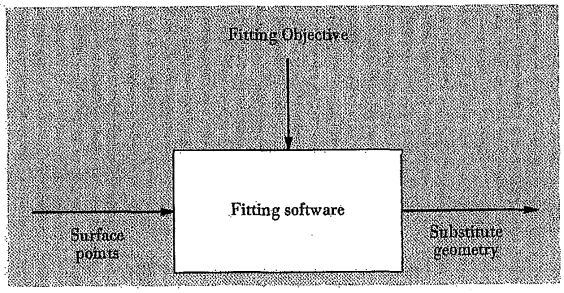

Fig. 1. Ideal model of fitting.

To date, testing methods have focused on the quality of software implementation [9]. These methods have been designed on the premise that the internal structure of the software is a "black box." Testing is based on evaluating the fitting results for specially designed data sets.

Published testing methods have been uniformly based on what might be called an external view of the behavior of the software: Is the software computing what it is advertised to compute? This view has two weaknesses. First, it provides few guidelines for designing a suite of data sets or for interpreting the results. Second, it provides no guidelines for evaluating the utility of different fitting objectives for an application. In this article we outline the elements of a software testing theory that is based on an operational model of error sources in fitting software. We believe that an operational model will provide a basis for designing test methods that are based on sound metrological principles.

The next section of this article discusses the relationship between fitting objectives, measurement error, and tolerance theory. The following section examines implementation issues: the interaction of optimization methods and computing environment, handling of extreme cases, and code correctness. The last section discusses the testing methods under development at NIST. We conclude with a summary of issues that need further attention.

FITTING OBJECTIVES

Fitting can be viewed as an optimization problem: We must find the parameters of substitute geometry that optimize a particular fitting objective for a set of points. The fitting process is illustrated in Fig. 1. As mentioned above, the choice of fitting objective is an important determinant of CMS performance. Part tolerances are

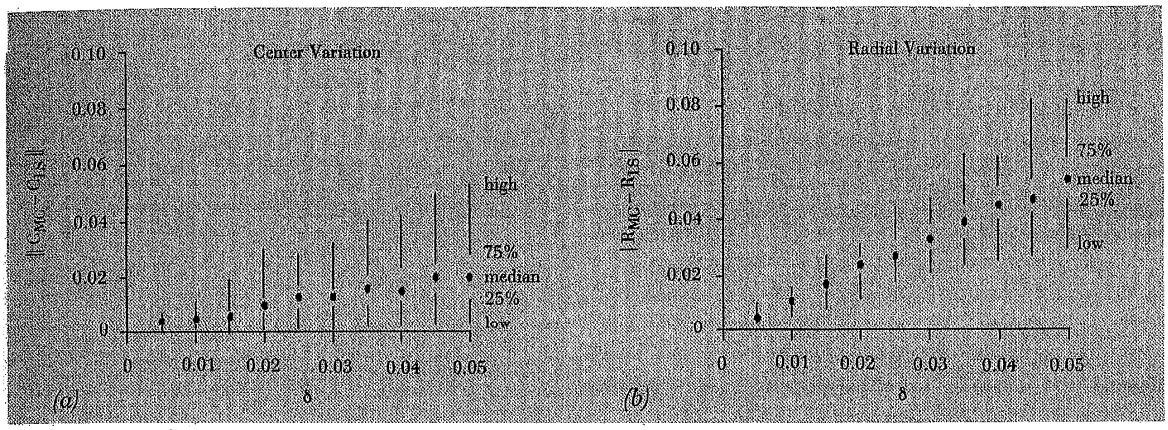

Fig. 2. Quartile plots comparing 100 least squares and minimum circumscribed fits for each value of δ.

generally interpreted in terms of *extremal fits*. That is, the fitting objective is to find the geometry that fits the extremes of the data: the largest inscribed, smallest circumscribed, or minimum separation geometry. Also, simulation of functional gages (sometimes called soft functional gaging) can be interpreted as finding the maximum clearance (or minimum interference) solid model fit to the data. On the other hand, measurement practice most commonly involves *averaging fits*, typically least-squares (orthogonal distance regression) but also objectives such as least median of squares [10, 11] and other robust techniques for outlier detection.

In general, averaging fits commonly used in metrology are biased with respect to the extremal fit objectives suggested by tolerancing theory. That is, in the limit as the point density goes to infinity and the measurement error for each point goes to zero, averaging fits will be different from extremal fits.[2] On the other hand, it would seem that averaging fits are less sensitive to measurement errors than are extremal fits [12].

To study these effects for least-squares and minimum circumscribed fitting objectives we used a circle-fitting problem. We generated 20 evenly spaced points around a unit circle and then perturbed the points by moving each in a random direction by a random distance. The directions were uniformly distributed over the unit circle and the distances were uniformly distributed between zero and an error scale factor δ. Perturbations were uncorrelated between points. We varied δ from 0.005 to 0.05 and generated 100 data sets for each value of δ. For each data set, we computed both the least-squares and minimum circumscribed fit.

Figure 2 shows the results of these experiments. Figure 2a shows the distance between centers and Fig. 2b shows the difference in radii for the two objectives. The median distance between centers for the two fit objectives varies approximately linearly with the perturbation scale δ, with a slope of about 0.5. Similarly, the difference in radii between the two objectives varies linearly with a slope of about one. If the perturbations are viewed as form errors in the feature, these results suggest that the bias of least-squares fitting is about half of the form error. The perturbations can also be viewed as errors in measuring individual points; a similar analysis, comparing the fits to the unperturbed circle, then shows that minimum circumscribed fitting is about twice as sensitive to measurement error as least-squares fitting. More generally, extremal fits will propagate more of the point measurement error than least-squares fitting.

Many fitting criteria can be expressed as special cases of a general criterion called L_p-norm estimation. This is a mathematically convenient formulation for which there are many results [13]. The L_p-norm estimation problem is to find the fit parameters that minimize the L_p norm:

2. Depending on one's viewpoint, bias might be considered to be an error in the model. We reserve the term *model error*, however, to indicate the error resulting from picking a particular geometric form to fit to the data.

$$L_p = \left[\frac{1}{n}\sum_i |r_i|^p\right]^{\frac{1}{p}}$$

where r_i is the i^{th} residual and the sum is over n data points. Least-squares fitting corresponds to the case $p = 2$. The limit of L_p as p goes to infinity is the largest magnitude residual, so that the L_∞ problem is minimizing the maximum magnitude residual—that is, finding the minimum zone fit. Generally, the bias and sensitivity errors of the fit will vary with p. This relationship is shown in Fig. 3. As p increases, the sensitivity of the fit to point measurement error increases, but the bias with respect to the fit prescribed by tolerance theory decreases. The precise shape of the curves depends on the distribution of residuals, the geometry of the particular fitting problem, the configuration of measured points, and the measurement uncertainty of the points.

We see, therefore, that it is very difficult to develop general guidelines for the proper choice of fitting objective for a practical CMS. Least-squares fitting is widely used and debated, with many claiming that extremal fitting is better because it conforms to tolerance theory. In fact, the best choice of fitting objective is that which produces the smallest combined uncertainty in the result. How to make the best choice is not at all clear.

We can modify the ideal model of fitting shown in Fig. 1 to include the effects identified in this section of the article. The new model, drawn in Fig. 4, shows that

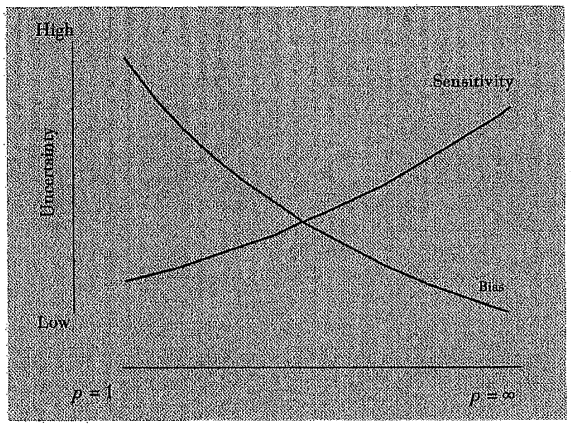

Fig. 3. L_p norm uncertainties.

the observable values—the data points being fitted and the reported fit—are combinations of inherently unobservable quantities. The observed data points always include some element of measurement error. The reported fit is a combination of the "true fit" (the fit to the surface points) and two additional errors: 1) the bias introduced by the choice of fitting objective and 2) the sensitivity of the fit to the measurement error.

The sensitivity and bias results reported here are highly specific. We need to develop more general results on these aspects of commonly used fitting objectives.

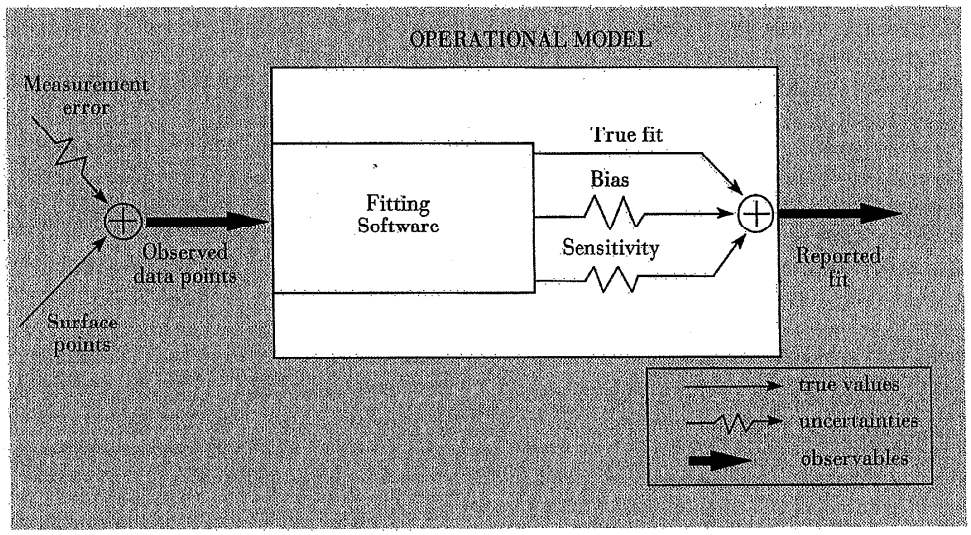

Fig. 4. Model of fitting software with bias and sensitivity errors.

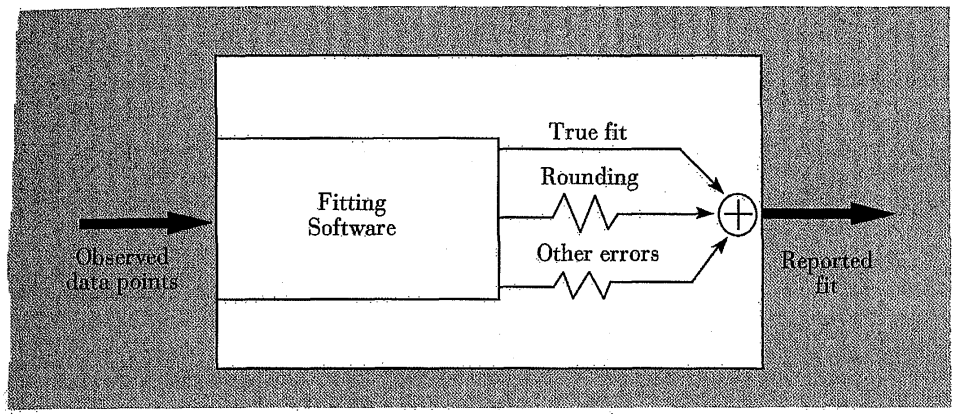

Fig. 5. The forward error analysis model.

The general results can then form the basis for practical criteria for selecting the best objective function for a particular application. Such criteria would be of great benefit in inspection planning.

Work is also needed to develop objectives with smaller combined uncertainty. For example, non-ideal geometry models should be used to reduce the model error of fitting perfect circles to features that are not circular. Going further, we would like to draw on pattern recognition techniques to develop methods for recognizing and classifying part form deviations. Finally, fitting objectives could make use of production process models. It is common knowledge in practice that information on how a part was manufactured will help an experienced inspector to do a better job. The inspector can draw on internal models of the form deviations expected from the manufacturing process to develop better inspection plans and data analysis strategies. Bayesian methods might be the basis for developing inspection systems with similar capabilities.

IMPLEMENTATION ISSUES

The second aspect of data analysis software performance discussed above is the implementation of the fitting objective in software. Four factors determine the quality of the implementation: 1) the optimization method chosen to compute the fit; 2) the computing environment (word length, rounding method, dynamic range, etc.); 3) handling of extreme cases; and 4) code correctness. Factors one and two, optimization methods and computing environment, are discussed in this section. Handling of extreme cases and code correctness are discussed in the following section on testing issues.

The choice of optimization method greatly affects how the computing environment impacts measurement uncertainty. In particular, the effects of rounding due to machine precision can be much greater for some optimization methods than for others, even for the same objective function. Analyses of rounding errors are often done in terms of a quantity called the unit roundoff error, denoted here by $\mathbf{u}$. The $\mathbf{u}$ is the largest number that, when added to one in floating point arithmetic, produces an answer of one. (Typically, $\mathbf{u}$ is about 10^{-7} for single precision and about 10^{-15} for double precision.) Some iterative algorithms rely on convergence tests to end. The effects of the convergence tests can sometimes be modeled as an increased value of $\mathbf{u}$ for the computing environment.

The study of how optimization methods and rounding errors interact can draw on results in linear algebra perturbation theory. Although fitting problems are often nonlinear in nature, most optimization algorithms work by repeatedly solving a linear approximation to the fitting objective. So the results of linear algebra can potentially be applied to a much broader class of optimization methods than might be apparent. (All results in this section are based on linear algebra found in [14].)

One example of how the choice of optimization method can dramatically affect the impact of rounding errors is simple linear regression. Suppose we wish to compute a regression line, represented by a vector v of slope and intercept, to the three points (–100, –100), (0, 0), and (100, 100). Clearly, the regression line has a

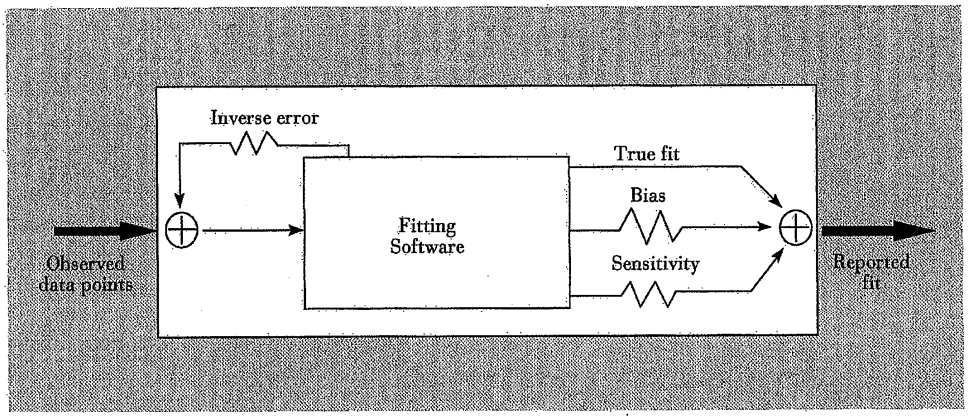

Fig. 6. The inverse error analysis model.

slope of one and a y-intercept of zero. Generally, the computed value of v (by any method) will be different from the exact solution v_{LS}. The fitting objective is to find the v that minimizes the sum of the squares of residuals. This can be modeled as solving for v in the matrix equation $Av \approx y$, where

$$A = \begin{bmatrix} -100 & 1 \\ 0 & 1 \\ 100 & 1 \end{bmatrix}$$

and $y = (-100, 0, 100)^T$. The most common solution method is the use of the *normal equations* for the model, obtained by multiplying the original matrix equation on the left by A^T and solving for v using matrix inversion. The result is $v = (A^TA)^{-1}A^Ty$. If v is computed with roundoff error $\mathbf{u}$ using the *Choleski decomposition* of A^TA, then the error of the solution is $\| v - v_{LS} \| \sim 6{,}700\mathbf{u}$. (The coefficient 6700 rises from the *condition number* of the matrix A^TA. The condition number roughly characterizes how hard the problem is. It is a measure of how close the matrix is to a singular matrix, which cannot be solved.)

The same objective can be evaluated using the singular value decomposition. A can be decomposed into the product $U^T\Sigma V$, where U and V are orthogonal matrices and Σ is a diagonal matrix. Let Σ^+ be the diagonal matrix obtained from Σ by setting each element of Σ^+ to zero if the corresponding element of Σ is zero, or else to the inverse of the corresponding element of Σ. Then $v = V\Sigma^+U^T$ is the least-squares solution. If v is obtained by evaluating the singular value decomposition with roundoff error $\mathbf{u}$, then the error of the solution is $\| v - v_{LS} \| \approx 82\mathbf{u}$. Thus, for this very simple problem, the effects of rounding can be changed by nearly two orders of magnitude by the choice of optimization method. (In this case, the measure of difficulty is the condition number of A, a much smaller number.)

The above is an example of forward error analysis. It states that the computed fit is the exact fit plus some error. This is illustrated graphically in Fig. 5. General results for forward error analysis can be obtained for some algorithms. For instance, orthogonal distance regression of a plane to a set of points can be solved using the singular value decomposition of the data matrix D. This algorithm has a relative error of about $\mathbf{u}\| D \|_2/(\lambda - \mu)$, where $\| D \|_2$ is the largest variance of the data in any direction, λ is the smallest variance of the data in the plane of the fit and μ is the variance of the residuals. Forward error analysis has the shortcoming that the relative error can only be characterized in terms of the same quantities (λ and μ) that are used to compute the fit. Thus, it is very difficult to write software for estimating the fit uncertainty that will work reliably in the same computing environment that is being used to calculate the fit.

A different approach, called inverse error analysis, views the computed solution as the exact solution to a nearby problem as illustrated in Fig. 6. Consider again orthogonal distance regression computed by the singular value decomposition of the data matrix D. The computed fit is the exact fit to some other data set, $D + \Delta$. Here, now, it is easy to characterize the effects of round-

ing; we have $\| \Delta_2 \| \approx c\mathbf{u} \| D \|_2$, for some constant, c, near one. The effects of rounding are modeled as adding more noise to the data before the fit is computed. The computation, however, is then considered to be done in exact arithmetic. The effects of the inverse error on the reported fit are determined by the sensitivity of the objective function.

Two methods, forward and inverse error analysis, can be used to study 1) the effects of the optimization method and 2) rounding on the uncertainty of reported fits. Only inverse error analysis results are known for many linear algebra procedures. This is because many of the factors that affect sensitivity also affect the forward error analysis, but need not be considered for inverse error analysis.

In principle, error analyses can readily be done for linear least-squares and total-least-squares problems solved by common methods. However, much more work needs to be done. Error analysis results need to be developed for all commonly used algorithms, and the analysis methods need to be extended to nonlinear problems. (These are often solved using linear approximations, so the extensions should be relatively direct). Harder, but equally important, is extending the methods to nondifferentiable objectives (such as extremal fits). Often, combinatorial methods are used for these objectives instead of iterative linear approximations. The ultimate objective for this work is the development of practical algorithms for estimating the uncertainty of a reported fit.

TESTING ISSUES

The purpose of testing is to assess the performance parameters of data analysis software in relationship to the theory of computational metrology, outlined in the previous two sections. Performance parameters include the error models identified above, the handling of extreme cases, and errors in coding. The results discussed in this section are being used to develop a software testing service to be offered by NIST.

Relatively little work is being done in software testing. As mentioned in the introduction, Germany and Great Britain offer a test service for least-squares fitting routines. A joint British/German project intends to extend these services to other fitting objectives. In the U.S., ASME Working Group B89.4.10 on CMS Software Performance is developing a standard for characterizing and testing the performance of data analysis software. The test service to be offered by NIST will directly support this emerging standard. We do not mean to suggest that this is the only work on inspection algorithms. Most work, however, is in developing new kinds of software or usage guidelines.

Our work on software testing is based on three assumptions. The first is that tests should be designed and interpreted according to a well-defined error model. The models presented above identify four error sources: bias, sensitivity, rounding error in the fit, and induced error in the data from rounding. The two other sources of error, handling of extreme cases and code correctness, will be discussed below.

The second assumption holds that test results must be interpreted and reported in terms directly related to inspection tasks. Part tolerances generally require conformance of the part to tolerance zones. These zones are volumes or areas for which the part features or substitute geometry elements must observe certain constraints. This means that test results should quantify the uncertainties of geometric relationships computed by the software. Representation-specific results (e.g., parameter values) should be avoided.

The third assumption is that the software to be tested is a "black box." In other words, the internal structure of the software—the optimization method, code structure, computing environment, and so on—is unobservable. Thus, the testing method is limited to supplying the software with fitting problems and analyzing the fit results. The problem set and the analysis are designed to identify the elements of the error models.

Assuming black-box testing leads to the now-common architecture of the testing system shown in Fig. 7. The system consists of three components: a data generator, a set of reference algorithms, and the fit analysis. The data generator is driven by a test description, a high-level specification of the collection of data sets to be generated. The data sets are processed by the software under test to generate test fits. At the same time, the data sets may be processed by fitting software supplied with the testing system. The data sets, test fits, and reference fits are used to assess performance measures for the software under test. In the remainder of this section, we will discuss the contents of a test description and how fit results are analyzed in the NIST system.

A data set is generated from four classes of information:

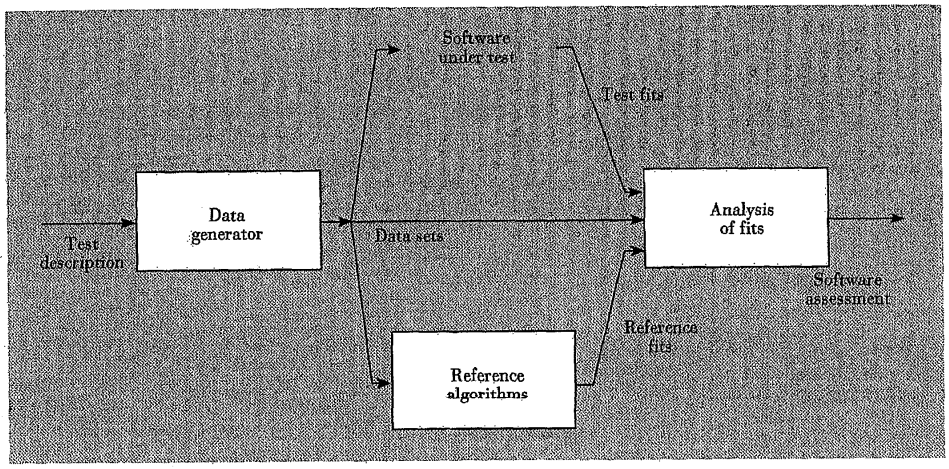

Fig. 7. Architecture of the NIST software testing system.

- the nominal (ideal) geometry of the feature;
- the form error(s) of the feature being simulated;
- the sampling plan (distribution of points on the feature) for the data set; and
- the (random) measurement error distribution for the points.

A test description consists of ranges of instance values for each of these information classes. This provides the flexibility to study the behavior of the software in a controlled manner. For instance, a test description that varies the simulated measurement error alone can be used to assess the sensitivity errors of the software for the particular combination of other factors. Similarly, variation of other factors can be used to study other aspects of the error model.

A comprehensive test will check the behavior of the software for extreme cases, which fall into three categories: pathologies, degeneracies, and extreme values. Pathologies are data sets for which no solution is possible. These include data sets with too few points and problems with unbounded solutions (such as fitting a circle to collinear points). Degeneracies are configurations that should pose no problem but often do. These include data sets with zero residuals (perfect geometry), "vertical" slopes, and similar configurations. Extreme values include number of points (minimum required; maximum of the implementation), distance from the origin, narrow geometry bounds (small arcs, etc.), and large residuals.

Code errors are difficult to detect. They can only be found if a test case exercises the relevant code. Code errors are detected in the analysis by comparing test fits to corresponding reference fits. This is, in fact, a primary purpose of computing reference fits. However, it is essentially impossible to design tests that will find all code errors in a black box. The best one can do is to generate data that are representative of the intended application.

As mentioned above, fits are analyzed in terms of geometric differences among fits, not parameter values. In the NIST system, all fit geometries are trimmed by the projection of the data onto the fit geometry. Comparison of test fits to reference fits is used to detect code errors and to identify systematic bias. Bias can arise when an implementation optimizes an approximation to the stated fitting objective. (Commercial software sometimes uses linear approximations in place of nonlinear objectives to gain a speed advantage.)

The main tool for evaluating software is measures of variation in fit geometry in response to varying test conditions. These measures are independent of reference fits. A different analysis is used for each type of geometry. For instance, cylinder fits are analyzed as follows.

The axis of each cylinder is trimmed by the orthogonal projection of its corresponding data set onto the axis.[3] The smallest cylinder that encloses the axis segments is then computed. The diameter of this

3. Fits from more than one data set may be analyzed together.

enclosing cylinder is a measure of dispersion for tolerances such as position, orientation, and parallelism. Depending on which parameters of the test description are involved, this dispersion is an estimate of sensitivity, bias, or effects of rounding errors. Similarly, the spread of radii of the cylinder fits measures dispersion for size. The spread of angles between the axes is measured as the angle of the narrowest cone that encloses all the direction vectors of the axes. This measure does not relate directly to common tolerance applications, but can be used for diagnostic purposes.

Comparison of fits to a reference is done on a data set-by-data set basis. (That is, a different reference is used for each data set.) Differences between test fits and the reference for each data set are computed in a manner similar to that described above. Thus, for instance, the maximum deviation of axis segments from the reference axis measures axis location differences. Difference measures can be accumulated across data sets to study the sensitivity of the differences to test parameters. A detailed description of analysis methods used at NIST will be reported elsewhere [15].

The theory of testing data analysis software is not well developed. The approach presented in this article is a start, but several research issues still need to be addressed. Software performance is affected by many factors, and it is a daunting task to define tests that represent all possible variations of these factors. Also, tests should be representative of the applications for which the software is to be used. Statistical design of experiments can be used to greatly reduce the number of tests, but appropriate models of the factors and their interactions need to be developed. Even a tightly designed test suite is likely to have too many cases to examine individually. A significant research challenge is the development of methods for summarizing and visualizing test results in ways that relate to the application and do not hide important information. The measures discussed above for dispersion and difference from reference fits are one form of summarization. They are useful for developing general conclusions, but there is no assurance that all important aspects of performance will be represented in the summary results. More work is needed in this area as well. Finally, the relationship of testing to the error models has only been developed in outline. Much work remains in understanding how to design and interpret tests that provide an accurate picture of software performance.

SUMMARY AND DISCUSSION

This article has discussed three aspects of computational metrology:

- fitting objectives;
- implementation issues; and
- testing issues.

In regard to fitting objectives, clearly, different measurement applications can differ on measurement goals. For instance, part inspection for tolerance conformance should be based on geometric tolerancing semantics [16]. Part measurement for process control, however, might better be based on vectorial tolerancing semantics [17]. We described a number of research issues regarding fitting objectives.

Any assessment of measurement uncertainty must be based, foremost, on a definition of the measurand. Only then can the bias of the fitting objective be assessed. The ASME Working Group Y14.5.1 on Mathematical Principles of Dimensioning and Tolerancing is developing semantics of design tolerances. This standard, expected now to be published in late 1994, will define the measurand for inspection applications.

A second aspect of computational metrology is the classification and study of implementation issues. The challenge is to develop effective operational models of software that support practical test methods. This paper has presented a model that appears to be effective for certain kinds of fitting. More work needs to be done in developing models for other fitting criteria. Accepted practice is that a measurement is incomplete without a statement of measurement uncertainty. Much remains to be done in developing tools for assessing measurement uncertainty in practical applications.

Data analysis software, like any tool, can be misused. Guidelines for proper use of such software are as important as models of software performance. A discussion of usage has been beyond the scope of this paper. However, the ASME Working Group B89.3.2 on Dimensional Measurement Methods is developing a standard that addresses this issue.

The third aspect of computational metrology is testing methods. We have described activity at NIST and elsewhere in software testing. The ASME B89.4.10

Working Group on CMS Software Performance is developing a standard for software testing. The planned NIST testing service will be based on, and will support, this standard. Today, there are no mechanisms to validate the results of data analysis with respect to international standards of length. As research issues are addressed and testing technology matures, we believe that concepts of traceability of software results will finally become possible.

Inspection software performance has become an important issue in the last few years and software problems, or their avoidance, have become a significant expense for many companies. Theories, tools, and standards in computational metrology are now beginning to emerge. It is our hope that with these advances, the utility of CMS software will greatly improve.

REFERENCES

1. ASME, ANSI/ASME Standard PTC 19.1-1985, Measurement Uncertainty, American Society of Mechanical Engineers, New York, 1985.
2. Porta C. and Wäldele F., Testing of Three Coordinate Measuring Machine Evaluation Algorithms, *Technical Report BCR EUR 10909 EN*, Commission of the European Communities, Luxembourg.
3. Walker R., *CIDEP Alert No. X1-A-88-01*, Government-Industry Data Exchange Program, Washington, DC, 1988.
4. Estler T., Accuracy Analysis of the Space Shuttle Solid Rocket Motor Profile Measuring Device, *NISTIR-89/4171*, National Institute of Standards and Technology, Gaithersburg, MD, 1989.
5. Feng S. C. and Hopp T. H., A Review of Current Geometric Tolerancing Theories and Inspection Data Analysis Algorithms, *NISTIR 4509*, National Institute of Standards and Technology, Gaithersburg, MD, 1991.
6. Cox M. G., Improving CMM Software Quality, *NPL Report DITC 194/92*, National Physical Laboratory, Middlesex, UK, 1992.
7. Wäldele F., Bittner B., Busch K., Drieschner R., and Elligsen R., Testing of Coordinate Measuring Machine Software, *Precision Engineering*, 15: 121–123, 1993.
8. Diaz C. and Hopp, T.H., Testing of Coordinate Measuring System Software, in the *Proceedings of the 1993 ASQC Measurement Quality Conference*, Gaithersburg, MD, October 27–27, 1993.
9. Cox M. G. and Forbes A. B., Strategies for Testing Form Assessment Software, *NPL Report DITC 211/92*, National Physical Laboratory, Middlesex, UK, 1992.
10. Rousseeuw P., Least Median of Squares Regression, *Journal of the American Statistical Association*, 79: 871–880, 1984.
11. Mamileti L., Wang C. M., Young M., and Vecchia D. F., Optical Fiber Geometry by Gray Scale Analysis with Robust Regression, *Applied Optics*, 31: 4182–4185, 1992.
12. Vecchia D. F., Wang C. M., and Young M., Outlier-Resistant Fitting of Gray Scale Images Illustrated by Optical Fiber Geometry, in the *Proceedings of the 1993 Measurement Science Conference*, Los Angeles, Jan. 21–22, 1993.
13. Gonin R. and Money A., *Nonlinear L -Norm Estimation*, Marcel Dekker, New York, 1989.
14. Golub G. H. and Van Loan C. F., *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1985.
15. Hopp T. H., Requirements for a Production Algorithm Testing System, National Institute of Standards and Technology, Gaithersburg, MD, in *preparation*, 1993.
16. Hopp T. H., The Language of Tolerances, *Quality through Engineering Design*, W. Kuo, Ed., Elsevier, New York, pp. 317–332, 1993.
17. Wirtz A., Vectorial Tolerancing for Production Quality Control and Functional Analysis in Design, *CIRP International Working Seminar on Computer-Aided Tolerancing*, pp. 77–84, 1991.

Theodore H. Hopp is a computer scientist at the National Institute of Standards and Technology in Gaithersburg, MD. His research interests include tolerancing theory, product data exchange, inspection methods, and software performance of coordinate measurement systems. He is involved in a number of national and international standards groups in these areas.